# Multivariable Linear Systems in SIMNON

Mårtensson, Bengt

1984

*Document Version:*
Publisher's PDF, also known as Version of record



*Citation for published version (APA):*
Mårtensson, B. (1984). *Multivariable Linear Systems in SIMNON*. (Technical Reports TFRT-7281). Department of Automatic Control, Lund Institute of Technology (LTH).

*Total number of authors:*
1

CODEN:LUTFD2/(TFRT-7281)/1-22/(1984)

# MULTIVARIABLE LINEAR SYSTEMS IN SIMNON

BENGT MÅRTENSSON

DEPARTMENT OF AUTOMATIC CONTROL
LUND INSTITUTE OF TECHNOLOGY
DECEMBER 1984

The simulation language Simnon is not too good at handling linear systems. External systems MIMOC and MIMOD has therefore been written. These are continuous time and discrete time multi input-multi output linear systems. Linear regulators, in which all parameters are available as inputs, are included. Therefore, the systems are very suitable for implementation of abstract adaptive controllers.

A single-input single-output system controlled by a relay has been implemented in the system RELAYSYS. A sensible logging feature is presented in the threshold sensitive logger TLOGGER.

The systems are intended to be very easy to use together with the matrix analysis package CTRL-C. A useful interface is presented, which is believed to some extent represent a new way of thinking.

LUND INSTITUTE OF TECHNOLOGY
DEPARTMENT OF AUTOMATIC CONTROL
Box 118
S 221 00 Lund Sweden

Document name
Report

Date of issue
December 1984

Document number
CODEN:LUTFD2/(TFRT-7281)/1-22/(1984)

Author(s)
Bengt Mårtensson

Supervisor

Sponsoring organization

Title and subtitle

Multivariable Linear Systems in Simnon.

Abstract

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
22

Recipient's notes

Security classification



Distribution: The report may be ordered from the Department of Automatic Control or borrowed through the University Library 2, Box 1010, S-221 03 Lund, Sweden, Telex: 33248 lubbis lund.

# Multivariable Linear Systems in Simnon

Abstract:
The simulation language Simnon is not too good at handling linear systems. External systems MIMOC and MIMOD has therefore been written. These are continuous time and discrete time multi input-multi output linear systems. Linear regulators, in which all parameters are available as inputs, are included. Therefore, the systems are very suitable for implementation of abstract adaptive controllers.

A single-input single-output system controlled by a relay has been implemented in the system RELAYSYS. A sensible logging feature is presented in the threshold sensitive logger TLOGGER.

The systems are intended to be very easy to use together with the matrix analysis package CTRL-C. A useful interface is presented, which is believed to some extent represent a new way of thinking.

## 1. Introduction

It is well known that the simulation language Simnon is not too good at handling linear systems. In a sense this is fairly natural: Simnon was made with a totally general nonlinear structure in mind, and thus it does not know e.g. the concepts of vectors and matrices. However, there are several situations when it is desirable to include linear systems in whole or in part in a simulation system. Despite of the fact that we from a mathematical point of view "can solve" e.g. the linear equation $\dot{x} = Ax + Bu$; a numerical solution is often desirable. Also, the main interest in application independent adaptive control is to control an unknown plant of the type

$$\dot{x} = Ax + Bu \qquad x \in R^n;\ u \in R^m$$
$$y = Cx \qquad y \in R^p$$

with a controller of the type

$$\dot{z} = F(\lambda)z + G(\lambda)y \qquad z \in R^\ell$$
$$u = -H(\lambda)z - K(\lambda)y$$

where $\lambda$ is some parameter updated according to $\dot{\lambda} = f(\lambda,y,u)$ for some function f.

An external Simnon system, MIMOC, has been written. This contains a linear system ot the type above, and a parameter dependent regulator. The parameters are available to e.g. a Simnon connecting system. When issuing the SYST command, MIMOC will, if told to do so, automatically generate a dummy connecting system, and/or Simnon code for a linear system with the same dimensions. The communication is arranged by global Intrac variables.

If told to do so, MIMOC will at the end of the simulation generate a file with all the parameters in the system, including the parameters in the "adaptive regulator

after convergence". These are read into CTRL-C by issuing a single command.

There is also a discrete time version, MIMOD.

The system RELAYSYS contains an single-input single-output linear plant as MIMOC and MIMOD, and a relay as a regulator. Hysteresis, deadzone, and asymmetric amplitude are implemented. A time delay between the output of the plant and the relay can be easily implemented by using the Simnon standard system DELAY.

TLOGGER is a system containing a threshold sensitive logger: Every input to TLOGGER is associated with a threshold level, and every time this is crossed, logging occurs.

It will be assumed in the sequel that the reader has a working knowledge of Simnon, e.g. from [1] or [4]. Some knowledge of Intrac, which in some sense is a part of Simnon, is required. The only up to date source of information is [7]. CTRL-C is an interactive matrix oriented program package analysis and synthesis, described in [3]. The work presented in this report is intended very much to serve as an interface between the linear algebra world of CTRL-C, and the differential equation world of (nonlinear) adaptive controllers and differential equations.

The systems are written in Pascal and linked into Simnon as described in [5].

In section 2 notations are introduced, and all variables defined. Section 3 contains the "how to use" information for MIMOC and MIMOD. The next two sections describe RELAYSYS and TLOGGER. As an example, an implementation of an "universal regulator" is done in section 6. A description of the source code is given in section 7. The next sections contains some comments and further suggestions. Appendix 1 contains a summary of notation, variables, macros, etc. A full session is given in appendix 2, based on the example in section 6. Finally, the last two appendices contains two macros intended to support the systems.

## 2. Structure and Notations for MIMOC and MIMOD

Let the plant to be controlled be

$$\dot{x} = Ax + Bu \qquad x \in R^n ; u \in R^m ; n,m \geq 1 \qquad (1a)$$
$$y = Cx + Du \qquad y \in R^p ; p \geq 1 \qquad (1b)$$

and the controller

$$\dot{z} = Fz + G(y-r) \qquad z \in R^\ell ; \ell \geq 0 \qquad (2a)$$
$$u = -Hz - K(y-r) \qquad (2b)$$

where r is a p-vector of reference inputs. Alternatively, it can be considered as output disturbance or measurement noise. (1) and (2) are included in the system MIMOC. ('C' stands for 'continuous time'.) The elements of A,B,C, and D are made parameters in the Simnon sense. r and the elements of the matrices F,G,H, and K are considered as inputs to MIMOC. x,y,u,z are considered as outputs. $\|x\|^2$ ( = $x_1^2 + \ldots + x_n^2$) and $\|y\|^2$ are also available as outputs with the names NORM2X and NORM2Y. The dynamics has also been augmented to include the (truncated)

$L^2$-norms squared, available as a state in MIMOC, i.e.

$$L2nrm2x = \int_0^t \|x\|^2 \, dt$$

where t is the present time. L2nrm2y is defined analogously.

Simnon identifiers are assigned to the elements of the vectors and matrices using as few indices as possible. See Appendix 1 for the exact rules.

There is also a discrete time system MIMOD, where the 'D' stands for discrete time, described by

$$x(k+1) = Ax(k) + Bu(k) \qquad x \in R^n ; u \in R^m ; n,m \geq 1 \qquad (1a')$$
$$y(k) = Cx(k) + Du(k) \qquad y \in R^p ; p \geq 1 \qquad (1b')$$

and

$$z(k+1) = Fz(k) + G(y(k)-r(k)) \qquad z \in R^\ell ; \ell \geq 0 \qquad (2a')$$
$$u(k) = -Hz(k) - K(y(k)-r(k)) \qquad (2b')$$

This system is completely analogous to MIMOC, with the following exceptions: It is possible to change the time interval by altering the parameter (in Simnon sense) $\Delta t$ = DELTAT. This is by default = 1. The $L^2$ norms are replaced by $\ell^2$ norms.

In the sequel the word MIMOX will denote MIMOC or MIMOD.

## 3. How To Use MIMOX

It might be a good idea to read this section more or less in parallel with the summary in Appendix 1, and the sample session in Appendix 2.

### 3.1 Assigning Dimensions and the A,B,C,D - matrices

There is two different ways of assigning dimensions and values to the matrices A,B,C and D. If the global Intrac variable MIMO.FILEIN = 0 or is left undefined the dimensions n,m, and p are read from the global Intrac variables MIMO.N, MIMO.M, and MIMO.P. If any of these is undefined, a default value of 1 is used. The parameters in the matrices are then possible to change in the usual Simnon manner. Default values are given so that $(A)_{ij}$ = 1 if i - j = 1, 0 otherwise; $(B)_{ij}$ = 1 if i = j = 1, 0 otherwise; $(C)_{ij}$ = 1 if i = j = 1, 0 otherwise; $(D)_{ij}$ = 0. This corresponds to $(G)_{ij} = s^{-n}$ if i = j = 1, 0 otherwise.

If MIMO.FILEIN = 1 then n, m, p, and the matrices are read from the file ABCD.MIM. This is a text file formatted according to the following rules: First three lines are skipped. These may contain textual information. Then n, m, and p are read in that order. Three more lines are skipped. A is read row by row. It may spread out over any number of lines. Three more lines are skipped until B is read in the same fashion. Etc... Observe that a D matrix has to be present, even if it is 0.

This file is intended to be generated from CTRL-C in the following manner: First define the matrices A, B, C, and D. While in CTRL-C you can do some fancy analysis of the system [A,B,C,D]. Then use the macro TOMIMO, included in Appendix 3. This will generate the file ABCD.MIM, properly formatted according to the rules above. A test is first made so that the dimensions are compatible.

### 3.2 Connecting System

A major idea behind these systems is the possibilities to make an (adaptive?) regulator for the system just by some simple lines in a connecting system, or eventually with some additional dynamics. See the example in the next section.

MIMOX aids in this: if you put the global Intrac variable mimo.conngen = 1, MIMOX will, when the SYST command is given, generate a dummy connecting system called MIMOCONN. This assigns all the inputs to the system to Simnon parameters, whose names are the names of the inputs, preceded by a "p". The main use of this code generating facility is believed to be as follows: There are often a fairly large number of inputs to MIMOX. MIMOCONN is a good starting point to edit to what you want, especially since it contains the right number of "wires", with the right names. Cf. the use in Appendix 2.

It is good practice to change the name and to remove the MIMOX signature when you edit the file. Again, cf. Appendix 2.

### 3.3 The Code Generation

If you put the global Intrac variable MIMO.CODEGEN = 1, MIMOX will when the SYST command is given, generate a continuous / discrete time system MIMO, with the same dimensions and parameters as itself. If MIMO.INCLNORM = 1 the Euclidean and the $L^2$ norms of both x and y will be included. If MIMO.INCLREG = 1 a regulator [F,G,H,K] will be included.

### 3.4 The Simnon Macro MIMOHELP

The Simnon macro MIMOHELP, presented in Appendix 4, aids in setting up the relevant global Intrac variables described above. This issues a question - answer dialogue and prompts the user for the values, expressed in the notation in section 2, or in plain English. Appendix 2 contains an example of its use. The author has found it very helpful to use, at least at initialization, instead of wasting time on remembering "what the ... was that called??" (After initialization, the Intrac WRITE command displays all assigned global Intrac variables.)

### 3.5 The Interface From MIMOX to CTRL-C

MIMOX contains a parameter FILEOUT, by default = 0. If it is > 0.5 MIMOX will, after completing a simulation, write the matrices A, B, C, D, F, G, H, and K (or all of them that are present) onto the file FGHK.MIM. This is a text file, with format similar to ABCD.MIM: First three lines are skipped, then the matrix follows etc. The skipped lines are used to visually separate the matrices, and to print their names. In the top of the file MIMOX prints its "signature", and the time the file

was generated.

In this case, MIMOX will also generate a file FROMMIMO.CTR containing a CTRL-C macro. In CTRL-C, this macro will read all the matrices from the FGHK.MIM file. This is also shown in Appendix 2.

## 4. The system RELAYSYS

Since RELAYSYS is written in the same spirit as MIMOX, this section is "incrementally written", i.e. it is not intended to be understandable without the prior sections.

RELAYSYS is a single-input single-output system, controlled by a relay. It is tailored after the report [2], and the notations are mostly from there.

The notations and variables are whenever possible the same as in MIMOX. D is assumed = 0, and omitted. m = p = 1, and the corresponding variables are also omitted. x,u,y,r,A,B,C,n,filein,conngen have the same meanings as before. TOMIMO, MIMOHELP, and ABCD.MIM are replace by TORELAY, RELAYHEL, and ABC.REL respectively. The generated connecting system is called RELAYCON.T.

The relay output levels are d1 and -d2. The hysteresis zone is $\epsilon$. There is also a deadzone. These are available as the parameters D1, D2, EPSILON, DEADZONE. See [2] and Appendix 1.

A time delay between y and the relay is achieved in the following way: Put the global Intrac variable RELAYSYS.DELAY = 1. The error signal e = r - y is then replaced by e = r - yin, where yin will be a new input to the system. The time delay, implemented by the Simnon system DELAY, should then be connected between y and yin. To use DELAY for this application, put N1.DELAY = 0, N2.DELAY = 1, and SPACE.DELAY = 5000 (or another "big number"). For more information on DELAY, see [7]. The automatically generated connecting system RELAYCON will have been set up accordingly.

Since the macros RELAYHEL and TORELAY are completely analogous to MIMOHELP and TOMIMO, they are not included in the report.

## 5. The Threshold Sensitive Logger TLOGGER

The system TLOGGER has n inputs u1,...,un. n is read from the global Intrac variable TLOGGER.N. The inputs ui, i = 1,...,n, has with them associated threshold levels ti, available as parameters. When, for some i, ui passes through the threshold level ti, u1,...,un will be written on one line of the text file TLOGGER.LOG.

Apart from obvious uses such as logging zero crossings etc, this can be used e.g. in the following manner: If you want ui to cause no logging, but to be included in the log, put ti "very big". You may want to treat the simulation time in this way. Periodic logging can be achieved by defining a "time modulo something", e.g. in the connecting system, and then log on "time modulo something", associated with a feasible threshold level.

For initialization it is assumed that before the simulations starts all ui are below their associated threshold levels ti.

## 6. Example: Implementation of an Adaptive Regulator

As an example of the use of MIMOC we implement a sort of "universal" controller, further described in [6]. Consider the system (1), where p = m and D = 0. Assume that all its transmission zeros are in the open left half plane and that CB has its eigenvalues in the right half plane. Under these conditions the controller

$$u = -ky \qquad k \in R$$
$$\dot{k} = \|y\|^2$$

will stabilize the system in the sense that $x \to 0$ and $k \to k_\infty < \infty$ as $t \to \infty$. For a proof see [6].

For this concrete example, we use n = 3, m = p = 2, and according to above, we put $\ell$ = 0. The A, B, C, and D matrices used are shown in Appendix 2. The open system has one real pole with negative real part, and two complex conjugated poles with positive real part. The only zero is located in -1, and CB has its eigenvalues in 1 ± 2i.

In order to implement this regulator we only have to make the 2 x 2 matrix K equal to the squared $L^2$ norm (of y) times the identity matrix. With the notation of the system, let K11 = K22 = L2nrm2y, and K12 = K21 = 0. This is done by a very simple editing of the connecting system generated by "SYST".

The full session is given in Appendix 2.

## 7. Description of Programs

The program is written in the framework described in [5]. "maxindex", the limit for all dimensions, has fairly randomly been put to 15. If this needs to be altered, only this constant needs updating.

The declarations of types, variables, and external procedures for Simnon is available in a separate "%insert"-ed file. General functions and procedures utilized are separately complied. The corresponding external declarations are also inserted.

The source code is not included in this report.

## 8. Tips and Comments

If DK $\neq 0$ ordinary Simnon would complain about "algebraic loop detected". This is not the case when you are using MIMOX, instead the outcome of then simulation will be slightly stochastic. (More precisely, it will act as a time variable time delay.) It is up to the user to avoid this.

The parameter CRASH in MIMOX is used in the following way: if $\|x\|^2$ > CRASH the simulation will be stopped gracefully. CRASH should normally be put to a very large number, slightly less than the largest real number the computer can hold. This will prevent the program to crash due to numeric overflow when things go wild. When simulating "universal regulators" utilizing "Nussbaum functions" things may go really wild, even if the algorithm is guaranteed to stabilize the system in the end! See [6]. Also compare the example in [5], where this parameter is further discussed.

If you "DO TOMIMO" in CTRL-C there is no need to "exit" CTRL-C (which yields a store-file) if you only are interested in the A, B, C, and D matrices. Instead you may "quit" (which yields no store file).

The reader with knowledge about DELAY maybe wonders why not Hermite interpolation was used together with RELAYSYS. This has been tryed: For obscure reasons unknown to the author Simnon accepts the solution presented here, but complains about "algebraic loop detected" when the derivative dy is used.

Don't forget how useful the $-commands are in Simnon and CTRL-C: You can e.g. start Simnon without leaving CTRL-C and vice versa. If you are giving several VMS-commands inside Simnon or CTRL-C it is faster to first SPAWN.

## 9. Further Suggestions and some Perspective

This work is intended to show a bit of the possibilities of linking several of the different "program worlds" together. There are some different worlds presently: the linear algebra world, the (nonlinear) differential equation and difference world, the polynomial world (including polynomial and rational matrices), the computer graphics world, the algebraic formula manipulation world. Reasonable good program packages exist for these worlds, maybe with the exception of the polynomial world. It will be a very interesting task for the future to link these worlds together. (Towards a global understanding!!) It should be aimed at generating a sort of language, wherein a very high level of programming in an interactive environment, e.g. similar to LISP, should be possible. It is on this framework algorithms for e.g computer aided design of control systems should be built. It will thus be simple to combine methods from several different worlds, e.g. numerical methods, algebraic manipulations, and heuristic search.

## 9. References

[1] Åström, K J, A Simnon Tutorial, CODEN: LUTFD2/(TFRT-3168)/1-52/(1982), Department of Automatic Control, Lund Institute of Technology

[2] Åström, K J and Hägglund, T, Automatic Tuning of Simple Regulators, CODEN: LUTFD2/(TFRT-7269)/1-07/(1984), Department of Automatic Control, Lund Institute of Technology

[3] CTRL-C Users Guide, Systems Control Technology, Palo Alto, California

[4] Elmqvist, H, Simnon Users Manual, Report TFRT-3091, Department of Automatic Control, Lund Institute of Technology

[5] Mårtensson, B, Pascal Systems in Simnon, Report CODEN: LUTFD2/(TFRT-7278)/1-13/(1984), Department of Automatic Control, Lund Institute of Technology

[6] Mårtensson, B, PhD - thesis, to appear (sometime!)

[7] Simnon's 'help' utility

# Appendix 1

Summary of Notation, Macros, and Variables for MIMOX

Notations

x : state vector of the plant
u : input vector to the plant
y : output vector of the plant
Cdx : C dx/dt
Cnx : C x(t+$\Delta$t)
z : state vector of the regulator
r : vector of reference inputs
A,B,C,D : matrices in the plant, see (1)
F,G,H,K : matrices in the regulator, see(2)
norm2x : ||x||
norm2y : ||y||
L2nrm2x : squared $L_2^2$ norm of x, see section 2
L2nrm2y : squared $L^2$ norm of y

n : dimension of the state vector x $\geq$ 1
m : dimension of the input vector u $\geq$ 1
p : dimension of the output vector y $\geq$ 1
$\ell$ : dimension of the state vector z of the controller $\geq$ 0

Global Intrac variables

mimo.n : by "syst" n is put = mimo.n if defined, otherwise 1
mimo.m : by "syst" m is put = mimo.m if defined, otherwise 1
mimo.p : by "syst" p is put = mimo.p if defined, otherwise 1
mimo.l : by "syst" l is put = mimo.l if defined, otherwise 0
mimo.filein : if = 1 n,m,p,A,B,C, and D will be read from the file ABCD.MIM
mimo.codegen : if = 1 Simnon code will be generated in the file MIMO.T
mimo.conngen : if = 1 a connecting system file MIMOCONN.T will be generated
mimo.inclreg : if = 1 a regulator will be included in the file MIMO.T
mimo.inclnorm : if = 1 the $L^2$-norms of x and y will be included

Parameters

fileout : if > 0.5 the files FGHK.MIM and FROMMIMO.CTR will be generated
crash : the simulation will be aborted when norm2x > crash.
deltat : sample interval in MIMOD, default 1

CTRL-C macros

tomimo : generates the file ABCD.MIM with A,B,C, and D
frommimo : reads A,B,C,D,F,G,H, and K from the file FGHK.MIM

Simnon macro

mimohelp : sets up the global Intrac variables by a question-answer dialogue

Files generated

MIMO.T : see mimo.codegen
MIMOCONN.T : see mimo.conngen
ABCD.MIM : see tomimo
FGHK.MIM : see frommimo

Rules for generating Simnon identifiers

Let T be a n x m matrix. The following rules apply for the names of the components of T:
min(n,m) = 0 (i.e. T nonexistent) : no name
n = m = 1 (i.e. T scalar) : T
min(n,m) = 1 (i.e. row or column vector) : T1,T2,...,T9,TA,TB,...
min(n,m) > 1 : T11,...

## Summary of Notation, Macros, and Variables for RELAYSYS

### Notations

x : state vector of the plant
u : input to plant
y : output from plant
yin : if delay is used : the delayed y-signal
dy : dy/dt
r : reference input
A,B,C : matrices in the system, see (1)
norm2x : $\|x\|$
norm2y : $\|y\|$
L2nrm2x : squared $L_2^2$ norm of x, see section 2
L2nrm2y : squared $L^2$ norm of y

n : dimension of the state vector x $\geq$ 1

### Global Intrac variables

relaysys.n : by "syst" n is put = relaysys.n if defined, otherwise 3
relaysys.filein : if = 1 n,A,B, and C will be read from the file ABC.REL
relaysys.conngen : if = 1 a connecting system file RELAYCON.T will be generated
relaysys.delay : if = 1 delay of y is used

### Parameters

crash : the simulation will be aborted when norm2x > crash
d1 : "positive amplitude" of relay, default 1
d2 : "negative amplitude" of relay, default 1
epsilon : hysteresis zone of relay, default 0
deadzone : deadzone of relay, default 0

### CTRL-C macros

torelay : generates the file ABC.REL with A,B, and C

### Simnon macro

relayhel : sets up the global Intrac variables by a question-answer dialogue

### Files generated

RELAYCON.T : see relaysys.conngen
ABC.REL : see torelay

## Summary of Notation and Variables for TLOGGER

### Notations

u : input vector to the system
n : dimension of the input vector u $\geq$ 1

### Global Intrac variables

tlogger.n : by "syst" n is put = tlogger.n if defined, otherwise 1

### Parameters

t : vector of threshold levels corresponding to u

### File generated

TLOGGER.LOG : output from TLOGGER

# Appendix 2

Sample Session

This appendix describes as a sample session an implementation of the adaptive regulator described in section 4. The following convertions will be used: CTRL-C-prompts will be denoted by '[>' in the leftmost column, VMS-promts by '$', while Simnon prompts are denoted by '>'. Lines not starting with any of these characters are considered as comments or output from a program. A '"' signals that the rest of the line is comment. Inserted files will be delimited by lines.

```
$ ctrlc    "invoke CTRL-C
[> A = [-1 2 3;1 2 -3;1 3 2]
[> B = [0 0;1 2;-2 1]
[> C = [0 1 0;0 0 1]
[> D = [0 0;0 0]
[> do tomimo
[> quit
```

'do tomimo' has now generated the file ABCD.MIM which looks as follows:

---

```
NMP   =

    3.    2.    2.

A     =

   -1.    2.    3.
    1.    2.   -3.
    1.    3.    2.

B     =

    0.    0.
    1.    2.
   -2.    1.

C     =

    0.    1.    0.
    0.    0.    1.

D     =

    0.    0.
    0.    0.
```

---

Now we enter the Simnon version containing MIMOC and MIMOD:
(we assume that you have it under the name "simnon" on the present directory)

```
$ invoke simnon    "Not 'run simnon', see [5]
```

```
> mimohelp  "This macro call issues the following question - answer dialogue:

Read from file? 0/1
> 1
Enter l
> 0
Do you want Simnon code generated? 0/1
> 1
Do you want a regulator included in the Simnon code? 0/1
> 1
Do you want norms included in the Simnon code? 0/1
> 1
Do you want a connecting system generated? 0/1
> 1
```

This has assigned values to the global Intrac-variables in the following manner:

```
mimo.l = 0
mimo.filein = 1
mimo.conngen = 1
mimo.codegen = 1
mimo.inclreg = 1
mimo.inclnorm = 1
```

The variables mimo.n, mimo.m, and mimo.p are left unassigned, since n, m, and p are read from the file.

```
> syst mimoc mimoconn  "There does not have to be present a file named mimoconn
```

Now MIMOC has generated the following Simnon code in the file MIMO.T:

---

```
continuous system MIMO

"  This file generated by MIMOC at 17:01:23.05   11-DEC-1984

state x1 x2 x3 L2nrm2x L2nrm2y
der dx1 dx2 dx3 dL2nrm2x dL2nrm2y
output norm2x norm2y
time t
zero = 0

dx1=a11*x1+a12*x2+a13*x3+b11*u1+b12*u2
dx2=a21*x1+a22*x2+a23*x3+b21*u1+b22*u2
dx3=a31*x1+a32*x2+a33*x3+b31*u1+b32*u2

y1=c11*x1+c12*x2+c13*x3
y2=c21*x1+c22*x2+c23*x3

norm2x=x1*x1+x2*x2+x3*x3
norm2y=y1*y1+y2*y2

dL2nrm2x = norm2x
```

```
dL2nrm2y = norm2y


u1=-k11*y1-k12*y2
u2=-k21*y1-k22*y2


a11 : -1.00000E+00
a12 :  2.00000E+00
a13 :  3.00000E+00
a21 :  1.00000E+00
a22 :  2.00000E+00
a23 : -3.00000E+00
a31 :  1.00000E+00
a32 :  3.00000E+00
a33 :  2.00000E+00

b11 :  0.00000E+00
b12 :  0.00000E+00
b21 :  1.00000E+00
b22 :  2.00000E+00
b31 : -2.00000E+00
b32 :  1.00000E+00

c11 :  0.00000E+00
c12 :  1.00000E+00
c13 :  0.00000E+00
c21 :  0.00000E+00
c22 :  0.00000E+00
c23 :  1.00000E+00



k11 :  0.00000E+00
k12 :  0.00000E+00
k21 :  0.00000E+00
k22 :  0.00000E+00

x1 : 0
x2 : 0
x3 : 0

end
```

---

and the following connecting system in the file MIMOCONN.T:

---

```
connecting system mimoconn

"  This file generated by MIMOC at 16:55:59.02   11-DEC-1984
```

```
time t
zero = 0



K11[MIMOC] = pK11
K12[MIMOC] = pK12
K21[MIMOC] = pK21
K22[MIMOC] = pK22

r1[MIMOC] = pr1
r2[MIMOC] = pr2



pK11 :  0.00000E+00
pK12 :  0.00000E+00
pK21 :  0.00000E+00
pK22 :  0.00000E+00

pr1 :  0.00000E+00
pr2 :  0.00000E+00

end
```

---

The connecting system is easily edited to read:

---

```
connecting system app2

zero = 0
k = L2nrm2y[MIMOC]

K11[MIMOC] = k
K12[MIMOC] = 0
K21[MIMOC] = 0
K22[MIMOC] = k

r1[MIMOC] = 0
r2[MIMOC] = 0

end
```

---

By this editing, we have really defined your adaptive regulator.

```
> let mimo.conngen = 0
> let mimo.codegen = 0

> syst mimoc app2
```

A 'disp' command now yields the following

```
------------------------------------------------------------------------------

                      CONTINUOUS SYSTEM mimoc
 STATE :  x1         0.00000E+00 x2         0.00000E+00 x3         0.00000E+00
          L2nrm2x    0.00000E+00 L2nrm2y    0.00000E+00
 INIT  :  x01        0.00000E+00 x02        0.00000E+00 x03        0.00000E+00
          L2nrm2x0   0.00000E+00 L2nrm2y0   0.00000E+00
 DER   :  dx1        0.00000E+00 dx2        0.00000E+00 dx3        0.00000E+00
          dL2nrm2x   0.00000E+00 dL2nrm2y   0.00000E+00
 INPUT :  K11        0.00000E+00 K12        0.00000E+00 K21        0.00000E+00
          K22        0.00000E+00 r1         0.00000E+00 r2         0.00000E+00
 OUTPUT:  y1         0.00000E+00 y2         0.00000E+00 u1         0.00000E+00
          u2         0.00000E+00 norm2x     0.00000E+00 norm2y     0.00000E+00
 PAR   :  A11        -1.0000     A12         2.0000     A13         3.0000
          A21         1.0000     A22         2.0000     A23        -3.0000
          A31         1.0000     A32         3.0000     A33         2.0000
          B11        0.00000E+00 B12        0.00000E+00 B21         1.0000
          B22         2.0000     B31        -2.0000     B32         1.0000
          C11        0.00000E+00 C12         1.0000     C13        0.00000E+00
          C21        0.00000E+00 C22        0.00000E+00 C23         1.0000
          D11        0.00000E+00 D12        0.00000E+00 D21        0.00000E+00
          D22        0.00000E+00 crash      1.00000E+10 fileout    0.00000E+00

 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

                      CONNECTING SYSTEM app2
 VAR   :  zero       0.00000E+00 k          0.00000E+00

------------------------------------------------------------------------------
```

```
> par fileout : 1
> store x1 x2 x3 k
> simu 0 10
> switch graph on
> ashow x1 x2 x3 k
```

The screen now looks like in figure 1.

At the end of the simulation, the file FGHK.MIM was generated:

```
------------------------------------------------------------------------

   This file generated by MIMOC at 17:03:09.77   11-DEC-1984
A        =

 -1.00000E+00  2.00000E+00  3.00000E+00
  1.00000E+00  2.00000E+00 -3.00000E+00
  1.00000E+00  3.00000E+00  2.00000E+00

B        =

  0.00000E+00  0.00000E+00
```

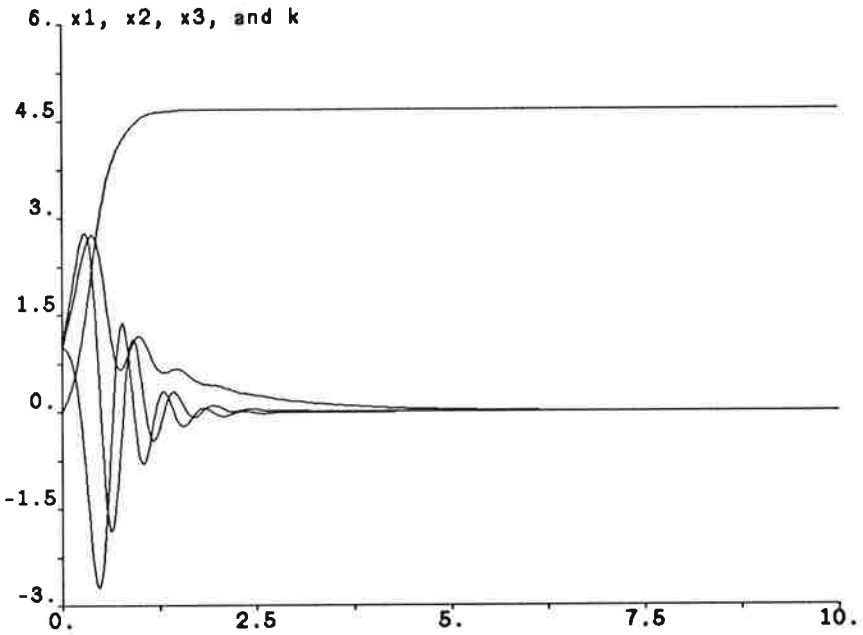


Figure 1 Outcome of the simulation

```
  1.00000E+00  2.00000E+00
 -2.00000E+00  1.00000E+00

C         =

  0.00000E+00  1.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  1.00000E+00

D         =

  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00

K         =

  4.68284E+00  0.00000E+00
  0.00000E+00  4.68284E+00
```

---

The following CTRL-C-macro, tailored for reading the file above, was also generated in the file FROMMIMO.CTR

---

```
// frommimo
//
```

```
// This crtlc-procedure will load the
// regulator matrices from MIMOC
//
// This file generated by MIMOC at 17:03:10.13   11-DEC-1984
//
load A <fghk.mim -f -s3 -r3 -c3
load B <fghk.mim -f -s9 -r3 -c2
load C <fghk.mim -f -s15 -r2 -c3
load D <fghk.mim -f -s20 -r2 -c2
load K <fghk.mim -f -s25 -r2 -c2
```

---

```
> $spawn      "generate a subprocess
$ ctrlc
[> do frommimo

Now you have access to all matrices in the system and regulator.

[> eig(A-B*K*C)

CTRL-C replies:

ANS =

   -2.7515 +12.1719i       "Stable, as we expected
   -0.8626 - 0.0000i
   -2.7515 -12.1719i

[> quit
$ logout   "Terminate the subprocess and return to Simnon
> .....
```

# Appendix 3

Tomimo

```
//  tomimo
//  This ctrlc-procedure dumps the matrices A, B, C, D
//  and their dimension to the file abcd.mim.
//  A check is made so that the dimensions are compatible.
//
error = 0;
[n1A,n2A] = size(A);
[nB,mB] = size(B);
[pC,nC] = size(C);
[pD,mD] = size(D);
if n1A <> n2A, error = 1;
if n1A <> nB, error = 1;
if n1A <> nC, error = 1;
if mB <> mD, error = 1;
if pC <> pD, error = 1;
if error = 1, tx = 'Incompatible dimensions'; display(tx);

if error = 0, nmp = [nB mB pC]; print nmp A B C D >abcd.mim;
```

# Appendix 4

Mimohelp

```
macro mimohelp
write 'Read from file? 0/1'
read mimo.filein int
if mimo.filein eq 1 goto nmpdef
write 'Enter n'
read mimo.n int
write 'Enter m'
read mimo.m int
write 'Enter p'
read mimo.p int

label nmpdef
write 'Enter l'
read mimo.l int

write 'Do you want Simnon code generated? 0/1'
read mimo.codegen int
if mimo.codegen ne 1 goto connsyst
write 'Do you want a regulator included in the Simnon code? 0/1'
read mimo.inclreg int
write 'Do you want norms included in the Simnon code? 0/1'
read mimo.inclnorm int

label connsyst
write 'Do you want a connecting system generated? 0/1'
read mimo.conngen int

end
```